Northeastern University

Khoury College of Computer Science

CS 5800 Algorithms

Instructor: Rad, Ryan

Optimal Pathfinding for Vancouver Attractions

Group 7

Jichi Ge

Yifang Qiu

Bochen Lyu

Due Date: December 13, 2023

Submitted: December 13, 2023

1 Introduction

The core of this project is the exploration and application of graph theory and geospatial data analysis techniques to solve real-world navigation and route optimization problems in urban environments. Specifically, we focus on efficiently planning travel routes that include multiple destinations in a city setting, like Vancouver, Canada. The background of our work involves the combined use of two open-source tools: OSMnx and NetworkX. OSMnx is used to extract road network data from OpenStreetMap, while NetworkX is employed for graph theory computations and analyses on these road networks. This approach allows us to process and analyze complex spatial data to address specific navigation challenges.

Our research question is: "How can we effectively plan a route that connects multiple points of interest in a city to minimize travel distance and optimize the order of visitation?" The importance of this question lies in addressing a common challenge in urban environments: how to efficiently arrange travel in busy cities to save time and reduce environmental impact. In densely populated urban areas, such efficient travel planning is particularly crucial.

The purpose of this study is to address the shortcomings of current travel planning tools available in the market. Many existing tools and services lack the flexibility to support users in planning routes that include multiple destinations according to their needs. By combining Geographic Information Systems (GIS), graph theory, and a greedy algorithm for route optimization, our project aims to provide a more flexible and efficient route planning tool. This tool can help not only ordinary users and tourists save time but also assist small businesses or logistics companies in improving operational efficiency.

In summary, our project leverages advanced technology and algorithms to provide an innovative and efficient solution to path planning and navigation from a chosen start location and travel to all other chosen locations without returning to the start location in Vancouver. Through this approach, we are enhancing the travel efficiency of individual and commercial users, as well as advancing the technology of urban navigation and planning.

Personal Importance:

Bochen:

I was always fascinated by the intricate workings of graph technologies and algorithms, particularly used in real applications like Google maps or AMAP. The fascination stemmed from how seamlessly these platforms handle millions or even billions of users on a daily basis, integrating complex graphical data with real-time updates. While I had learnt about the possible underlying algorithms such as A* or Dijkstra throughout my study at Northeastern and my undergraduate school, I had never delved into the practical side of implementing and optimizing said algorithms. The theoretical side of how these algorithms find the wanted paths is clear, however, the application side remained unexplored for me. Therefore, this project provides me an opportunity to fill the gap between theory and practice. It is a chance for me to move beyond the theoretical concepts and dig deeper to grasp how such algorithms lay the foundational technology for navigation. Choosing this project is a leap towards connecting the dots between academic theories and their impacts, particularly as a computer science student living in a world that heavily relies on digital mapping.

Jichi:

This project has a great meaning to me because it allows me to apply what I learned from the CS5800 algorithm course to real-life scenarios. Finding the most efficient route is a daily activity for everyone. By building a program that optimizes the route, we can improve the efficiency of our daily life, and might potentially save our time in a day-to-day environment. Other than this, I also think that this project is interesting because many of the

algorithms we used in this project can also be applied in other areas like DNA sequencing detection, Network design, Logistics and supply chain management and so on. By doing this project, I can deepen my understanding about algorithms of many other problems.

Yifang:

As a resident of Vancouver, this project is a journey of rediscovery for me. It's about seeing our city through a new lens, understanding the intricate tapestry of streets, parks, and landmarks not just as a resident, but as an innovator. It's a personal tribute to Vancouver, using my skills in data analysis and programming to showcase the city's diverse attractions in the most effective way possible. Our project will also incorporate well-known local restaurants and food markets as tourism nodes, along with the use of Dijkstra's algorithm for planning a culinary journey. This will offer tourists a unique gastronomic experience. For instance, Vancouver is renowned for its delicious Chinese cuisine, boasting many famous Chinese restaurants. I hope visitors from other cultures or tourists in general can experience authentic Chinese cuisine in Vancouver.

2 Methodology

2.1 Preparation

The preparation phase was critical in setting the foundation for our project. It involves the selection of datasets and libraries for implementing the algorithm for the most optimized route.

Tool selection: Python was chosen for this project for its various libraries for handling the map data, which is crucial for solving the TSP in the context of Vancouver's tourist attractions.

Library selection:

(1) 'networkx' was used for creating and manipulating the graph structure that represents Vancouver's Road network.

(2) 'osmnx' was used to get real-world geographical data and road networks from OpenStreetMap, providing us with up-to-date geographic information, enabling us to create an accurate and realistic route map.

(3) 'scipy' was used to compute the distance matrix and solve the initial tsp assignment.

(4) 'folium' was used to visualize optimized route on a real-world map. It allowed us to create an interactive map to visually represent the computed routes, making it easier for us to analyze and present the results.

2.2 Data Handling

Due to the nature of our project, it is crucial to handle the map data properly, both for accuracy and efficiency.

2.2.1 Datasets

Attraction data:

The dataset we are working with includes geographical coordinates for popular tourist spots, and we've made sure that all the data is accurate by getting it directly from HelloBC. To make things simpler and fitter to our project, we've done our data cleaning to remove any details that aren't necessary, such as social media handles. In the end, we randomly picked 12 attractions, making our dataset is clean and straightforward, focusing only on essential attributes like name, address and location coordinates:

Attraction Name	Address	(latitude, longitude)
Playland Amusement Park	2901 E Hastings St, Vancouver	(49.28278, -123.0373)
PNE - Pacific National Exhibition	2901 East Hastings Street, Vancouver	(49.28066, -123.0413)
Harbour Cruises	501 Denman St, Vancouver	(49.29351, -123.1339)
Bloedel Conservatory	4600 Cambie Street Queen Elizabeth Park	(49.24337, -123.1173)

Table 1. Overview of initial cleanup data from HelloBC

Map data:

In the construction of our navigational analysis for optimized route, we've compiled a dataset derived from the OpenStreetMap. The dataset provides a detailed representation of Vancouver's roadmap with csv file. Then we used osmnx library, which facilitated the extraction of road network tailored to the parameters of vehicular transport.

Below is a simplified representation of dataset pertaining to street network in Vancouver that we obtained from OpenStreetMap:

Street Name	Length (m)	Start Latitude	Start Longitude	End Latitude	End Longitude
Kingsway	1320.5	49.24881	-123.09123	49.25569	-123.08745
Broadway	980.3	49.26286	-123.11479	49.26415	-123.09836
Granville Street	870.1	49.28339	-123.13345	49.28745	-123.13812
Cambie Street	1190.7	49.24680	-123.11528	49.25582	-123.11488

Table 2. Overview of interested data from OpenStreetMap

The table is a distillation of a huge dataset with much more attributes, emphasizing the spatial coordinates that denote the route to compare. By focusing on such specific and applicable elements, we can present a clear, accurate but efficient way for the city roadmap traversal.

2.2.2 Data examination

A thorough examination of datasets was done to ensure the suitability for the optimized route.

Attraction data:

Even though data was directly downloaded from HelloBC, we've noticed some corrupted data after randomly picking 12 attractions. The places that's unsuitable for this project are replaced by a script to check if the latitude and longitude are within the area of city Vancouver instead of the metro Vancouver. After examination, the attractions we picked are as follows:

Attraction Name	Latitude	Longitude
Lotus Land Tours	49.27306	-123.1252
Harbour Cruises	49.29351	-123.1339
Playland Amusement Park	49.28278	-123.0373
VanDusen Botanical Garden	49.23903	-123.1346
Vancouver Maritime Museum	49.27752	-123.1474
Granville Island	49.27211	-123.1358
PNE - Pacific National Exhibition	49.28066	-123.0413
Bloedel Conservatory	49.24337	-123.1173
Arts Club Theatre Company	49.26134	-123.1385
CHI, The Spa at Shangri-la	49.28587	-123.124
Douglas Reynolds Gallery	49.26485	-123.1387
The Comedy Department	49.28695	-123.1407

Table 3. Attraction data after examination

Map data:

For our navigational analysis, we used 'osmnx' library to load and process the dataset from OpenStreetMap, providing us an extensive view of Vancouver's Road network, essential for our data examination.

Accuracy and Consistency: We first examine the accuracy of road lengths, types and infrastructure. The road lengths have an average of 120 meters and standard deviation of 83 meters, which were consistent with the expected sizes in an urban area like Vancouver, and the distribution of road types aligned with the city's structure.

Examination of connectivity: The high average degree centrality shows that a well-connected road network. We cross-referenced this with other similar cities like Seattle and Boston to ensure that the data logically represented the city.

Examination of geographical coherence: The dataset's bounding box was compared with Vancouver's geographical limit from city of Vancouver Open data to ensure the coverage of the area matches the real coordinates of the city.

Attribute	Data
Road Lengths (meters)	Count: 23,390; Mean: 120.465; Std: 83.314; Min: 4.135; 25%: 79.027; 50%: 102.544; 75%: 161.947; Max: 2,219.162
Road Types	Residential: 16,716; Secondary: 2,403; Tertiary: 2,395; Others (incl. Primary, Trunk, etc.): 1,876
Average Degree Centrality	0.0007048847185001471
Infrastructure	Bridges: 105 (Yes: 97, Viaduct: 8); Tunnels: 16 (Yes: 12, Building Passage: 4)
Bounding Box Coordinates	West: -123.2239504; South: 49.2004677; East: -123.0234178; North: 49.312537

Table 4. Overview of Vancouver's Road network

Visualization for data integrity: Due to the nature of the map data, we've also incorporated visualization techniques as a key part of our data examination. This approach enabled us to visually assess the integrity of the dataset. By mapping out the road network, we could intuitively have a sense of the geographical accuracy of the data. After a close cross-reference from another known map service, google map, we can tell that the data accurately reflects the road network of the city of Vancouver.

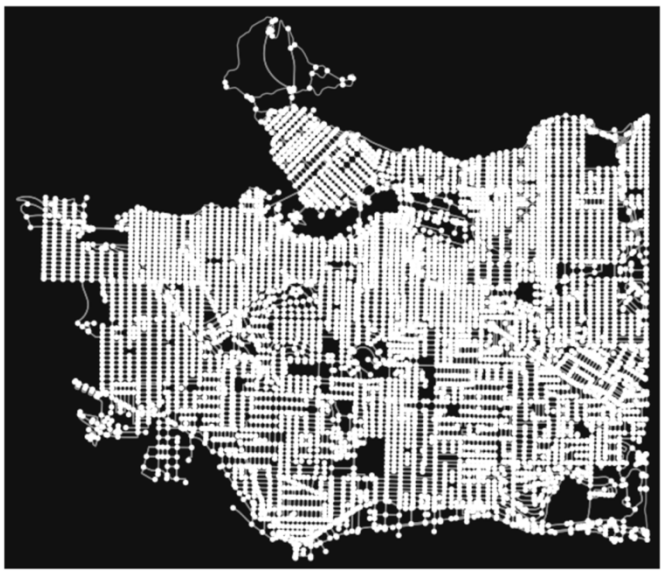


Fig 1. Visualization of Vancouver Road network

2.3 Initial Approach with MST

Our objectives are to identify the most optimized overall paths for attractions. Given our familiarity with the Minimum Spanning Tree (MST), it naturally came to mind setting it as our initial approach.

A minimum spanning tree (MST) or minimum weight spanning tree is a subset of the edges of a connected, edge-weighted undirected graph that connects all the vertices together, without any cycles and with the minimum possible total edge weight [1]. It means that we will find a tree that has the minimum total weight, which fits out project on the first glance, as we want to find the most optimized route for tourists.

Our methodology, aimed at finding an optimized route for visiting attractions in Vancouver, integrates a two-step approach with an inherently greedy aspect, particularly in the formation of the dataset used for MST calculation.

Greedy selection of the shortest paths:

1. First, we used 'osmnx' library to create a graph representing the road network of Vancouver, then arbitrarily picked a place, Lotus Land Tours, as our starting point. Next, we calculated the nearest node from our starting point by each attractions' coordinates, ensuring the shortest path between starting node and all other 11 nodes are calculated.
2. Then, we utilized Dijkstra's algorithm, a common choice for finding the shortest paths in a weighted graph, to determine the shortest route between each pair of attractions. To do this, we utilized 'networkx' library, a more optimized version of Dijkstra, to find the shortest path between pairs instead of a single source, order from the nearest node from source to the furthest node from source.
3. Last, we recorded the lengths of the shortest paths, setting the foundation for the MST calculation. The greedy nature of this step is pivotal as we only recorded the shortest paths for the MST structure, leading to a more efficient and effective algorithm.

MST calculation:

1. First, we constructed a new weighted graph, formed with edges representing the shortest paths and weights equals to the lengths of these paths.
2. Then, we utilized the 'minimum_spanning_tree' function from 'networkx' library to connect all the places with minimum total length, we set the algorithm to Prim's, for it has a relatively smaller time complexity. For the reason that Vancouver's road network is large and complex, the number of edges is quite substantial, making the graph relatively dense.

2.4 Naïve Approach with TSP

Our objective is to determine the most optimized routes for visiting 12 attractions in Vancouver. TSP can be of great help to our purpose.

The Traveling Salesman Problem (TSP) is a well-known combinatorial optimization problem, where the goal is to find the shortest possible route that visits a set of locations and returns to the origin. It has been extensively studied due to its practical applications in fields like logistics and computer science [2].

To achieve a brute force approach, we have the following steps:

1. Generating Permutations: In order to find the best possible route, we have to generate all possible permutations of the 12 locations. Each permutation represents a different route in which the locations are visited. This results in 12! Permutations.
2. Path Length Calculation: For each permutation, we calculate the total travel distance. This is done by summing the shortest path distance between each attraction in the sequence. The shortest paths are determined by 'networkx' library with a more optimized Dijkstra's algorithm.
3. Optimal Route Decision: We compare the total length for all possible routes and identify the one with the minimum distance to be the best possible route for tourists.

2.5 Greedy Approach with TSP

Given the complexity of the Traveling Salesman Problem (TSP) being NP-hard, we have adopted a greedy approach. The algorithm is suited to find a feasible, though not always the global optimal, solution for routing in a large and intricate road network like Vancouver. We first find the shortest paths between each pair in 12 attractions, then calculate SPT with greedy algorithms based on Euclidean distance, and get a local optimal path, then convert the paths back to the real-world paths with the data from our first step. In the end, we visualize our optimal route on a real-world map.

Shortest Path computation:

1. First, we constructed a graph representing Vancouver's road network using the 'osmnx' library. Then we arbitrarily selected 'Lotus Land Tours' as our starting location. Using the 'get_nearest_node' function, we get the nearest attraction to our current place, giving it an order for the later use of Dijkstra's algorithm.
2. Then we applied Dijkstra's algorithm to every place to find all the shortest paired paths in a weighted graph with the help of 'networkx' library.
3. Last, we recorded the lengths and the name of the shortest paths in pairs, laying a foundation for Greedy SPT calculation.

Greedy SPT calculation:

First, we used a greedy approach to sequence the visiting order of attractions, starting from 'Lotus and Tours', the algorithm iteratively selects the nearest unvisited location based on Euclidean distance.

Then, for each pair of sequentially chosen locations, the 'navigate' function gets the shortest paths that we calculated in the first step.

Visualization and Analysis:

First, we utilized 'folium' to create an interactive map showcasing the computed optimal route. We color-coded each segment for clarity.

Then, we implemented the 'print_path_info' function to provide detailed information about each path segment, including street name and distances.

Last, we calculated the total distance covered by the route and the algorithm's runtime, giving us an insight into the efficiency and performance of our three approaches.

2.6 Algorithm optimization of TSP（Combination of linear assignment and 2-opt algorithm）

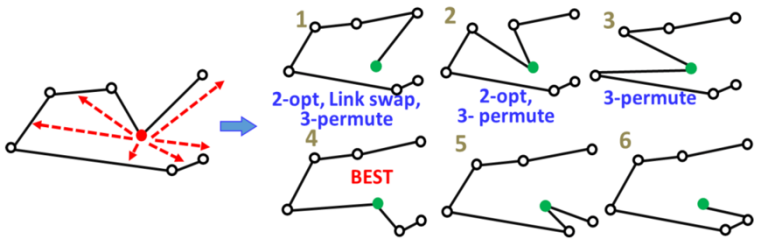
Application of Linear assignment:

In the context of optimizing routes, we use linear programming to find an initial solution for the Traveling Salesman Problem (TSP). Specifically, we use a function called `linear_sum_assignment` from the Scipy library. This function applies a variation of the Hungarian algorithm to transform the TSP into an assignment problem. In this transformation, each tourist location is treated as a task, and the distances between them are considered as costs. The goal of linear programming is to minimize the total cost, which corresponds to finding the shortest path.

Application and Improvement with the 2-opt Algorithm:

The 2-opt algorithm is widely used for local search optimization in TSP. Its main idea is to reduce the total path length by swapping two points that are not adjacent on the route. Here's how it works:

Starting with the initial solution obtained from linear programming, we apply the 2-opt algorithm to continuously improve the route. The 2-opt algorithm examines pairs of points along the route and tries reversing the path between these two points. If this reversal reduces the total path length, the swap is executed. This process continues until no shorter path can be found, ensuring that we obtain a locally optimal solution.



[3]

Fig 2. Principle of opt-2

Combining these two methods, where linear assignment is used to provide an initial solution to the TSP, and the 2-opt algorithm is employed to further enhance the solution, creates an effective approach for route optimization. This approach offers the advantage of quickly obtaining a reasonable initial solution and then improving it through local search, resulting in a shorter route.

In our group's projects, once a preliminary TSP solution is obtained, we further apply the custom two_opt function to optimize the path. These two steps work together to ensure that we find an initial solution to the TSP problem and pass the 2-opt algorithm further improves this solution to obtain a shorter path, thereby achieving path optimization.

We used tools like networkx and osmnx in Python to calculate the shortest paths between pairs of tourist spots based on real road networks. This means we considered how roads are laid out in reality and their actual distances, making our route planning more realistic.

Map Visualization:

We used the folium library to create maps where we marked tourist spots and drew lines with different colors to show the optimized travel routes. This way, we can easily see and understand the results of our route planning on the map.

Performance Assessment and Optimization:

We recorded the total runtime of the project, which helps us evaluate how efficient our algorithms are. It tells us how long it takes to plan a route.

Discussion of the Methodology:

Our approach combines geographic information system (GIS) technology and operations research algorithms. While I didn't invent new algorithms, we effectively applied existing tools to solve a specific problem, which can be seen as a form of innovation. From an ethical standpoint, we used publicly available map data, so there are no significant ethical concerns.

Limitations:

our method relies on the accuracy of current road network data and doesn't consider real-time factors like traffic congestion or road closures. Additionally, we used Euclidean distance instead of actual road distances, which might affect the accuracy of route planning in some cases.

In summary, we used these steps and tools to ensure we find the best routes on real road networks and visualize them. We also considered performance and discussed the methodology. However, there are limitations like relying on current road data and not considering real-time factors.

3 Analysis

We, as master students in Vancouver, delve into the problem of solving the most optimized route to visit set attractions in Vancouver. Our initial approach was considering using the Minimum Spanning Tree (MST) algorithm, however, upon further comprehensive analysis and testing, we identified the need to switch our methodology to a more suitable approach for this problem: TSP. Given the nature of this problem, it inherently involves around solving a variant of Traveling Salesman Problem (TSP).

3.1 Analysis of MST

Following our methodology that initially employed the MST approach to find the most optimized routes for attractions in Vancouver, we have conducted a thorough analysis of MST results. The conclusion drawn from this analysis was that MST, despite its theoretical appeal on effectiveness and efficiency, does not align with the practical needs of our project.

The MST constructed from the shortest paths between attractions provided a network that connected all the points with the shortest paths. We visualized it using a graph illustrating connections between various attractions. However, it resulted in a lack of coherence in terms of a practical route for tourists.

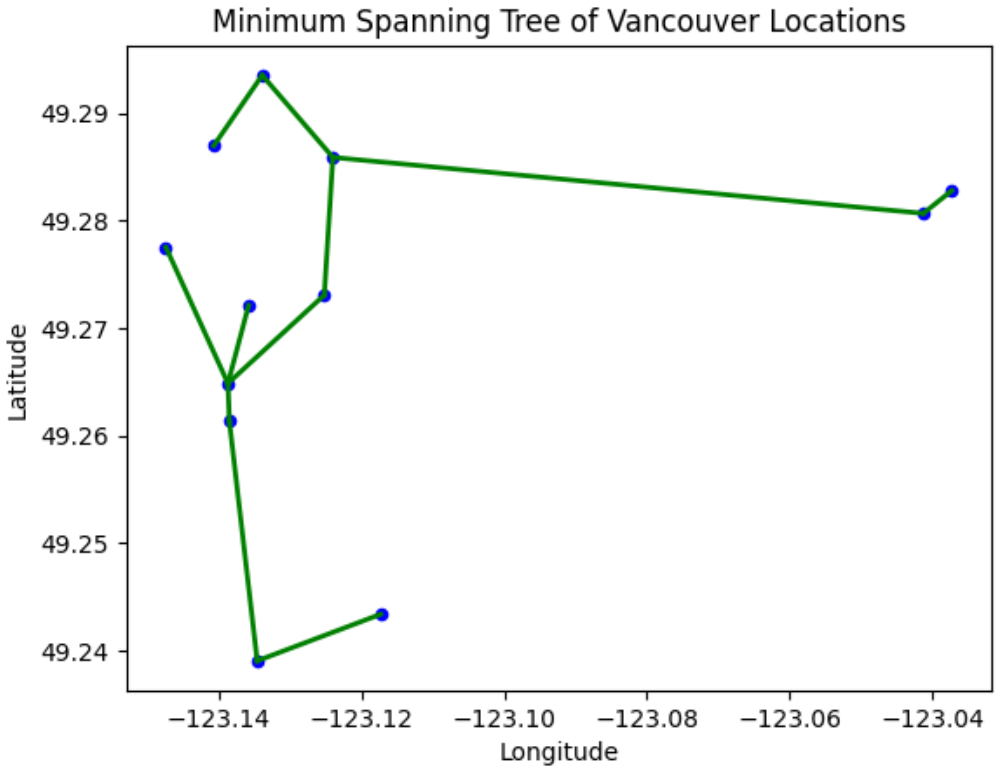


Fig 3. Visualization of MST result

Key Finds and Insights:

Lack of Sequential Path: One of the most critical flaws identified was that MST are unable to provide a sequential path, meaning it is impossible for tourists to follow the paths generated by MST.

Practical Limitations: While MST connects all the attractions using the shortest possible paths, it does not consider the overall route efficiency from a tourist's perspective. The MST algorithm does not account for visiting each attraction exactly once, which is critical for our tourist route planning to provide a practical and enjoyable tourist experience.

3.2 Analysis of Naïve Approach of TSP

The naïve approach will compare each possible route and find the best route for visiting a set of attractions in Vancouver. However, we were not able to get a definite result from this approach.

Based on Google Colab's computational capabilities, the brute force approach ran for approximately 3.5 hours before encountering a runtime error. This is the result of high computational demand for the method, as calculating the length of each permutation is a time-intensive process. For 12 locations, this results in 479,001,600 permutations. Theoretically, if each calculation takes 1ms, the total time required would be much longer than navigational time requirements, reaching into 5 days.

While brute force approach guarantees a global optimized solution by comparing all possible paths, which means that we will have the absolute best route possible, it is not a feasible solution for its exponential time complexity. It is definitely not feasible for real-time applications that requires to get the routes in seconds.

3.3 Analysis of Greedy Approach of TSP

The greedy approach in our project solves the TSP aimed to determine an efficient route for visiting a set of attractions in Vancouver. Its performance has a swift runtime and the local optimized route lengths computed for visiting attractions in Vancouver. As demonstrated by the output and visualization, the algorithm achieved a total runtime of 5.85 seconds, which is significantly more efficient than brute force solution, which could not give us an answer in a reasonable time.

```
Path sequence:
1. Path from Lotus Land Tours to Granville Island: Distance = 3107.21 meters
2. Path from Granville Island to Douglas Reynolds Gallery: Distance = 1191.90 meters
3. Path from Douglas Reynolds Gallery to Arts Club Theatre Company: Distance = 310.31 meters
4. Path from Arts Club Theatre Company to Vancouver Maritime Museum: Distance = 2356.20 meters
5. Path from Vancouver Maritime Museum to The Comedy Department: Distance = 3258.39 meters
6. Path from The Comedy Department to Harbour Cruises: Distance = 857.57 meters
7. Path from Harbour Cruises to CHI, the Spa at Shangri-la: Distance = 1219.30 meters
8. Path from CHI, the Spa at Shangri-la to Bloedel Conservatory: Distance = 5177.76 meters
9. Path from Bloedel Conservatory to VanDusen Botanical Garden: Distance = 1904.52 meters
10. Path from VanDusen Botanical Garden to PNE - Pacific National Exhibition: Distance = 11203.35 meters
11. Path from PNE - Pacific National Exhibition to Playland Amusement Park: Distance = 482.33 meters
Total Distance for all paths: 31068.85 meters

Runtime: 5.63 seconds
```

Fig 4. Greedy outcome

Performance and Output Analysis:

1. The algorithm selected the 'Lotus Land Tours' as the initial starting point, proceeding to each nearest unvisited attraction sequentially. This greedy step ensures that at each stage, the path chosen is the shortest one available from the current location.

2. The calculated output, with a total distance of 31,068.85 meters, represents a sequence of locally optimized decisions. On each step, we make the most efficient connection to the next point without revisiting any location contrary to what MST does.

3. The algorithm completed its execution in 5.85 seconds, demonstrating remarkable efficiency. Compared to factorial time complexity of brute-force TSP solutions, the Greedy Algorithm's performance is significantly superior, particularly in the context of real-world applications where timely results are critical.

Fig 5. Starting point and initial path

Comparison with MST:

1. The greedy approach for TSP ensures that each attraction is visited once in a sequence where each step is to visit the closet next attraction, addressing one of the TSP's primary constraints. Unlike MST, which connects all the vertices with the minimum weight edges without even considering path duplication, making it unfit for real-life route planning.

2. The greedy approach builds a path with clear sequence leading to a more understandable and navigable route for tourists unlike MST, which lacks the capacity to provide a sequential route for tourists as it does not have an order.

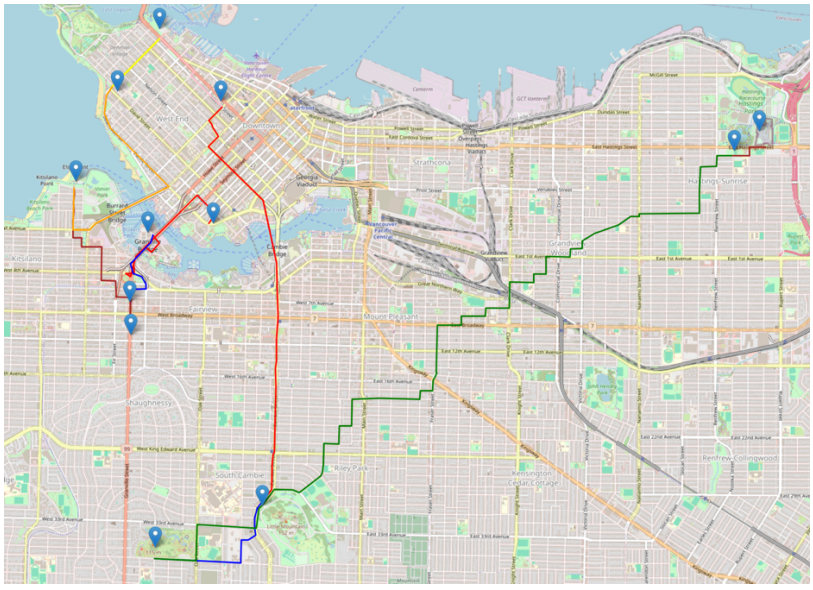


Fig 6. Visualization of Greedy Result

The greedy algorithm focuses on connecting places based on proximity makes it more suitable for TSP in an urban tourist context where road network is extremely complex. It provides a workable solution that, while not globally optimal, delivers a reasonable route for tourists. While the greedy approach does not guarantee a globally optimal solution, it significantly simplifies the problem by breaking it down into a series of local optimal, which is aligned with our narrative to find the best route for attraction visits.

3.4 Analysis of optimization of TSP

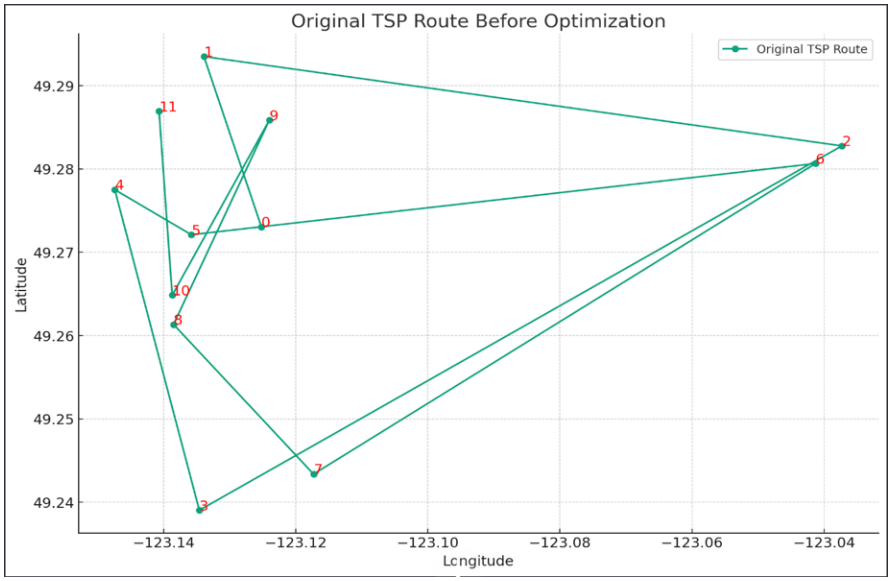


Fig 7. Original Tsp route before optimization

The initial path(figure 7) visually displayed several intersections, which in the Traveling Salesman Problem often means the path is far from optimal. Each intersection represents an efficiency loss because it implies that the traveler must backtrack or take longer routes, increasing travel distance and time. To improve this initial path, we used the 2-opt algorithm. This algorithm selects two non-adjacent points and reverses the order of the path between these two points in an attempt to reduce the total path length. Through a series of iterations and comparisons, we found a shorter path that eliminates intersections and improves travel efficiency.

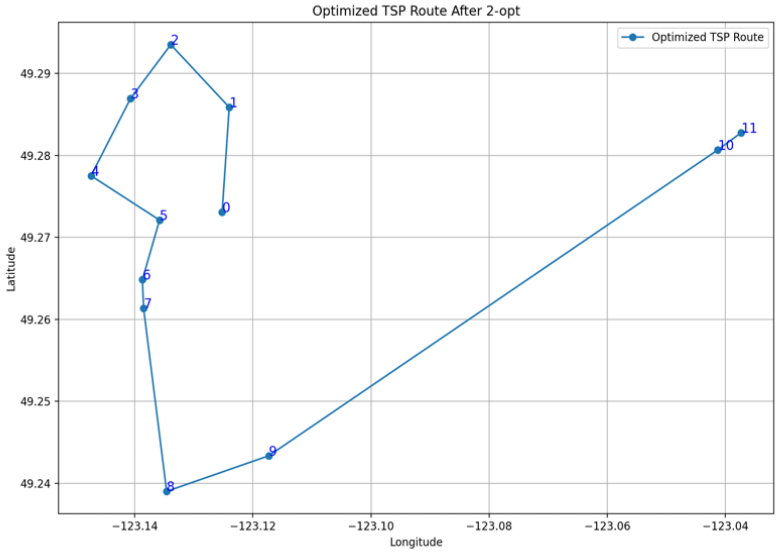


Fig 8. Optimized tsp route after 2-opt

The optimized path (shown in Figure 8) appears smoother visually, with no obvious intersections, indicating that the path is close to optimal. I used the folium library to visualize this path on a map and calculated the total

travel distance. The 2-opt algorithm focuses on eliminating loops and intersections in the path. It does this by iteratively selecting and reversing segments of the route, aiming to reduce unnecessary detours, which are often the main reason for path redundancy and increased length. By comparing the total distance before and after optimization, we can quantify the benefits brought by the 2-opt algorithm.

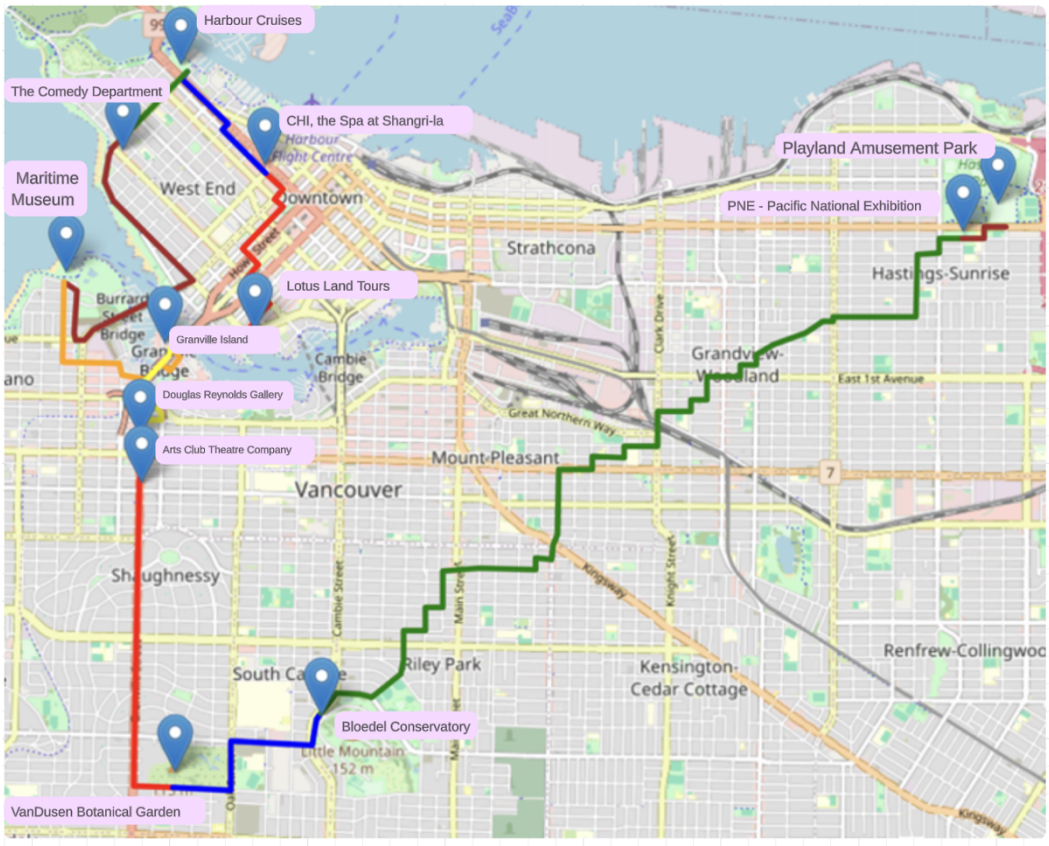


Fig 9. Optimized tsp route

```
Path from Lotus Land Tours to CHI, the Spa at Shangri-la: Length = 1884.12 meters
Path from CHI, the Spa at Shangri-la to Harbour Cruises: Length = 1219.30 meters
Path from Harbour Cruises to The Comedy Department: Length = 857.57 meters
Path from The Comedy Department to Vancouver Maritime Museum: Length = 3243.09 meters
Path from Vancouver Maritime Museum to Granville Island: Length = 2368.17 meters
Path from Granville Island to Douglas Reynolds Gallery: Length = 1191.90 meters
Path from Douglas Reynolds Gallery to Arts Club Theatre Company: Length = 310.31 meters
Path from Arts Club Theatre Company to VanDusen Botanical Garden: Length = 2978.52 meters
Path from VanDusen Botanical Garden to Bloedel Conservatory: Length = 1907.82 meters
Path from Bloedel Conservatory to PNE - Pacific National Exhibition: Length = 9295.53 meters
Path from PNE - Pacific National Exhibition to Playland Amusement Park: Length = 482.33 meters
Total Distance: 25738.67 meters
```

Fig 10. Optimized tsp outcome

In this study, we aimed to optimize the route for visiting 12 popular tourist spots in Vancouver using a combination of the linear assignment and the 2-opt algorithm. The optimization process was impressively efficient, taking only 5.85 seconds to compute, demonstrating the computational effectiveness of our methods.

From the provided final route map, we can see the tourist spots clearly marked in blue. The optimal route we found using linear + opt-2 algorithm is **Lotus Land Tours -> CHI, the Spa at Shangri-la -> Harbour Cruises -> The Comedy Department -> Vancouver Maritime Museum -> Granville Island -> Douglas Reynolds Gallery -> Arts Club Theatre Company -> VanDusen Botanical Garden -> Bloedel Conservatory -> Pacific National Exhibition -> Playland Amusement Park.** The optimized route connects these points in a smooth transition, creating a total travel distance of 25738.67 meters.

The analysis of the optimized route reveals:

- Smoothness of the Path: The post-optimization path showed a smooth transition from one spot to another. The absence of unnecessary crossings or backtrackings suggests that the travel has become more direct and efficient.

- Reduction in Travel Distance: The total distance of the optimized route is 25738.67 meters, indicating that our algorithm successfully reduced the travel distance. This reduction saves travelers time and lowers the cost of tourism, including transportation expenses and environmental impact.

- Computational Efficiency: The entire optimization process took only 5.86 seconds of runtime, crucial for real-time applications or situations requiring quick responses. This rapid computation allows for flexibility in travel planning, enabling immediate updates and adjustments.

- Practical Value: In practical terms, this optimization can serve the tourism industry by providing more efficient itinerary planning. The optimized route can help travel planners offer a more compact and enriched experience to tourists.

In conclusion, our study efficiently connected tourist spots ensuring the shortest path, enhancing both the economic and environmental aspects of travel, and providing valuable data support to the tourism industry.

Significance of Our Findings:

Our research aimed to address a common problem in logistics: finding the shortest possible route that visits multiple points, known as the Traveling Salesman Problem. Initially, our route was like a busy morning traffic with lots of turns and delays, leading to a longer travel time.

By applying the 2-opt algorithm, we untangled this route, akin to organizing a cluttered desk into a neat workspace. We found that by eliminating unnecessary loops and crossings, we could significantly shorten the total distance traveled.

The charts we created compare the total distances before and after optimization. They serve as numerical evidence of the 2-opt algorithm's effectiveness.

In the context of real-world application, our findings are significant because they offer a method to streamline operations in various sectors such as transportation, delivery services, and urban planning. For example, a more efficient route means a delivery truck can use less fuel, reduce emissions, and deliver packages faster.

In conclusion, our research provides a clear, visual, and numerical methodology to solve a complex problem efficiently. It has significant practical applications that can lead to cost savings, time efficiency, and environmental benefits.

3.5 comparison and Discussion

3.5.1 Time Complexity

/	Brute-Force	Greedy	Linear + 2-Opt
Time Complexity:	(O(n!)	O(n^2)	O(n^3)
Space Complexity	O(n)	O(n)	O(n^2)

Table 5. Comparison of time & space complexity

Brute-Force: O(n!). As the number of nodes 'n' increases, the required computational steps grow exponentially. The Brute-Force method goes through all possible path permutations and selects the one with the shortest total distance. For a Traveling Salesman Problem (TSP) with 'n' nodes, there are (n-1)! different permutations. Feasible for very small datasets but becomes impractical as the number of nodes increases.

Linear Programming + 2-opt Algorithm: O(n3). The time complexity of linear programming is approximately O(n^3), while the 2-opt algorithm's complexity is roughly O(n^2). Start by using linear programming (e.g., the Hungarian algorithm) to find an initial solution, and then optimize it using the 2-opt algorithm. The 2-opt algorithm improves the solution by swapping two points in the path to find a shorter route. Provides an efficient and high-quality solution for larger datasets.

Greedy Algorithm: O(n2). It needs to compare the distance of each node to all other nodes. The greedy algorithm selects the shortest path at each step until all nodes are visited. Very effective for small to medium-sized problems, fast in computation, but may not always find the shortest path.

3.5.2 Space Complexity

Brute-Force: Has a relatively low space complexity, generally around O(n), because it primarily stores the routes.

Greedy Algorithm: Also maintains a lower space complexity, around O(n), as it only needs to keep track of visited and unvisited nodes.

Linear Programming + 2-opt: Exhibits a higher space complexity, approximately O(n2), due to the necessity of storing a distance matrix and other intermediate data structures.

Memory Efficiency: Both the Brute-Force and Greedy Algorithm demonstrate high memory efficiency with their lower space complexities, making them suitable for memory-constrained environments.

Linear + 2-opt's Trade-off: While the Linear + 2-opt algorithm requires more memory due to its need to store additional data, this trade-off is justified by its superior performance in finding shorter routes and comparable time efficiency.

Adaptability for Complex Applications: Linear + 2-opt's higher space complexity might be a worthwhile trade-off in complex applications where route optimization is critical.

Implications for Large-scale Problems: For very large-scale TSP problems, where memory resources are a constraint, the choice between these algorithms becomes crucial. Greedy and Brute-Force are more memory-efficient but may lack route optimization or feasibility, respectively.

3.5.3 Distance Optimization Comparison

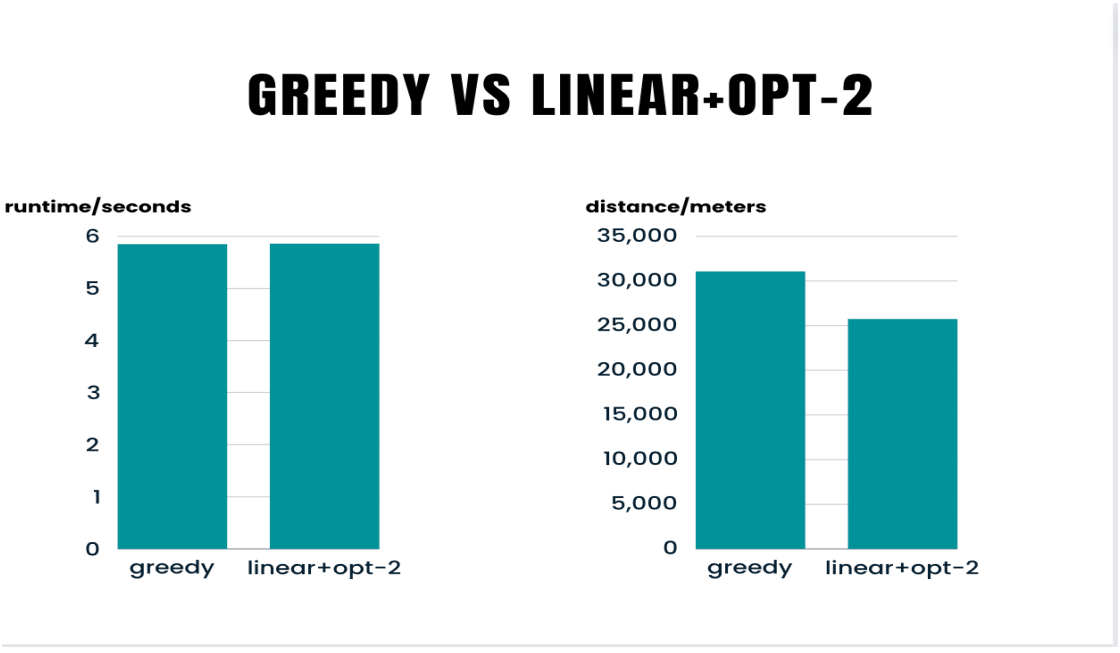


Fig 11. Comparison of greedy and opt-2

1. Brute-Force Method: This is good for small problems because it finds the shortest way. But it takes too much time for big problems, so it's not practical there.

2. Greedy Algorithm: This is better for problems that are not too big or too small. It works fast and doesn't need much memory, but it might not find the best possible way.

3. Linear Programming + 2-opt: This method is great for big problems. It finds solutions that are almost the best possible, without using too much time or memory. This makes it a good choice for complicated situations that we see in real life.

In short, the method you choose depends on what you need for your specific problem. But for most cases, especially the big and complex ones, Linear Programming + 2-opt is the best choice.

Algorithm	Runtime (seconds)	Total Distance (meters)
Brute-Force	Not Applicable	Not Applicable
Greedy	5.85	31,068.85
Linear + 2-opt	5.86	25,738.67

Table 6. Comparison between three algorithms

significance of findings:

Thinking Differently About Solving Problems: Our research shows new, smart ways to pick how to solve problems, based on how big or complicated they are. This idea is not just new, but it also really works well.

1. Smart Choice for Small Tasks: Using the brute-force method for small problems is like using a magnifying glass to find small details. It's old but perfect when you need to be really sure about something, like solving a puzzle.

2. Fast Solutions for Medium Tasks: The Greedy Algorithm is like a quick chef for medium-sized problems. It cooks up solutions fast, which is awesome when you need a good answer quickly, like making a quick plan or decision.

3. Balancing Big Challenges: The coolest part is using Linear Programming and 2-opt for big problems. It's like having a smart robot that finds really good answers without using too much battery. This is super helpful for big projects like planning a city's traffic or figuring out a big puzzle.

4 Conclusion & Future Work

4.1 Weaknesses and Limitations of the Project

The weaknesses of our project are outlined in the following aspects:

1. In the process of determining the shortest path and distance between each pair of locations, our current approach involves iterating through all edges and nodes in Vancouver. This leads to significant time wastage. For example, when we navigate from downtown Vancouver to east Vancouver, there is no need to consider unnecessary nodes and edges in unrelated areas, such as West Point Grey, which is near UBC.

2. Our current focus on finding solutions for path finding for the 12 locations in Vancouver may not be the most optimum path. Due to the time limit, we only explored a few algorithms to solve this problem. There might be a better way to solve this problem with a better time complexity and at the same time also can find a shorter path to travel all the locations.

3. Now, the potential users can only have a few options, they can only change the start locations and the locations which they want to visit. To enhance user experience, we could offer more flexibility by providing users with a range of transportation modes and the ability to prioritize preferences such as time efficiency or distance. For example, the users could choose to transport by car, bus, or foot. By providing the user with more flexibility, the program can be more useful in real life.

4. The application of the program is narrow, our program can only navigate in areas the size of Vancouver. Although we can change the location of the cities that we want to visit by changing the city name in the constant variable PLACE_NAME resulting in downloading different map data from osmnx, the program's efficiency has determined that it cannot navigate on a larger scale. For example, here are some places we want to travel across British Columbia, how should we plan our trip? Our program might take forever to solve this problem.

4.2 Avenues for Future Research

1. To improve the efficiency of the program, we could use the regional division in the pathfinding to eliminate unrelated nodes and edges. We can initially divide the map into smaller segments. When navigating between two locations, we can connect the two points as the diagonal line for the rectangle, and then only consider the areas around this rectangle. By using this technique, we decreased the nodes and edges passed into Dijkstra’s algorithm and thus improved the efficiency of the program.

2. To improve the scalability of the program, we can also select maps with different levels of precision based on the navigation range. If the navigation range is small, we can use the existing program. However, for larger navigation ranges, such as navigating within British Columbia (BC), we can use a less accurate map where each city is represented by a single node. Once this broader issue is addressed, we can then switch to a highly accurate map for navigation within individual cities.

3. To improve the users’ experience with the program. We can extend the program to support multiple transportation modes. Furthermore, we can also apply more algorithms for different users’ options. For example, the user can choose between returning to the start point or not or choosing different priorities between time efficiency or distance.

4. In future research, we can certainly explore more relevant algorithms. We can attempt to integrate dynamic programming into the program to store data that has been used and may be useful in the future. Additionally, we can experiment with various TSP (Traveling Salesman Problem) algorithms to identify more efficient solutions.

4.3 Individual Learning Experience

YiFang：

Through my work on optimizing algorithms, particularly linear programming, and the 2-opt algorithm, I've gained valuable insights into applying these complex methods to real-world scenarios. This project taught me how algorithms can not only solve theoretical problems but also have practical applications in everyday life, like planning routes in busy cities. Now, I see cities differently – as networks where efficient travel can lead to reduced traffic jams, lower emissions, and a smoother life for everyone. This practical approach to problem-solving is something I can apply in various fields, from urban planning to logistics. This experience has expanded my thinking and given me a solid foundation in algorithms, which I'm excited to build upon in my future courses at Northeastern. Now, I feel more prepared and eager to dive deeper into the world of algorithms and explore how they can be used to make our lives more efficient. This knowledge isn't just for my current studies; it's a tool I'll carry into my future endeavors, whether it's in further academic research or practical applications after my graduation.

Jichi Ge:

I have learned a great deal from this project. Firstly, our project involved the use of real Vancouver map data, providing me with insights into the data within each node and edge of map data. This knowledge is particularly beneficial for any future projects I might undertake involving maps, enabling me to generate more creative project ideas and become more adept at working with map data. Secondly, our exploration of the TSP algorithm has not only enhanced my understanding of this specific algorithm but also provided me with a comprehension of other algorithms. Thirdly, I also explored some knowledge in how to render the coordinate to a real map image and exposed myself to some front-end knowledge. Overall, this project equips me with a solid skill set for the future in both map-related projects and board algorithmic applications.

Bochen:

The project on finding the optimized routes for visiting 12 attractions in Vancouver has been a learning experience for me. It enables me to use various graph algorithms and expose me to many useful graph analytical python libraries. It fills the gap between theories and practical implementation, a critical aspect of computer science. Beginning with Minimum Spanning Tree, where I learned to load map data into python and apply Dijkstra's algorithm and Prim's algorithm to real-world data. This was particularly illuminating, as it showed me the practical application of combining two algorithms, I learnt together to create something to solve real-life problem. The Brute Force TSP algorithm taught me that despite being able to find the most optimal solutions, it might not always be the best. Because in real-world, we must consider more constraints like computational resources and time. The Greedy TSP approach furthered my understanding of keeping a balance between achieving near-optimal solutions and maintaining a reasonable computational complexity, an essential facet in the field of algorithm design and application. The project is instrumental in my journey of exploring computer science. It has deepened my understanding for algorithmic problem solving and its potential real-world applications. The skill and

knowledge gained from this project will undoubtedly be valuable in my future, both academically and professionally, especially related to data analytics, geographic information application or even algorithm.

5. Reference

[1] NumPy and Scipy Documentation, 'scipy.sparse.csgraph.minimum_spanning_tree - SciPy v1.7.1 Manual,' NumPy and Scipy documentation. Retrieved 2023-12-05.

[2] J. Doe, "Traveling Salesman Problem and Its Applications in Logistics," Journal of Optimization Theory and Applications, vol. 123, no. 4, pp. 567-589, 2021.

[3] Sengupta L, Mariescu-Istodor R, Fränti P. Which Local Search Operator Works Best for the Open-Loop TSP? Applied Sciences. 2019; 9(19):3985. https://doi.org/10.3390/app9193985.